Besides representing DFAs and NFAs using diagrams we can express them using their full definition. Similar to a Turing Machine, we can define the alphabet, $\Sigma$, the set of states, $Q$, and the set of transitions $\delta$. For problems 1 and 3 you will be defining an NFA and DFA respectively. Please submit your answer to these questions as a plain text files instead of pdfs.

Your automata should be formatted first by specifying the alphabet with the label $\text{sigma:}$ followed by a newline. Then on each line specify a symbol in the alphabet. After the last character in the alphabet you will specify the states. This is denoted $\text{Q:}$ and followed by a newline. Then each state is specified in the automata, one state name per line. The start state should be named $\text{qstart}$.

After the last state, you will specify the transitions in the automata. This is denoted $\text{delta:}$ and followed by a newline. Each line that follows is one transition specified as:

$\langle \text{statename}, \text{alphabetsymbol} \rangle \rightarrow \text{statename}$

Finally, the list of accepting states are given. This is denoted $\text{accept:}$ and followed by a newline and one accepting state per line. Note: for the epsilon transition in an NFA, we will use $\text{eps}$, this is not a member of $\Sigma$. A possible transition could be something like $\langle q4, \text{eps} \rangle \rightarrow q5$. You can also add comments to your code by using a #

So for the above example, the solution is:

$\text{sigma:}$

0
1

#This is a comment

$\text{Q:}$

$q\text{start}$
$q2$
$q3$
$q4$
$q5$
$q6$
delta:
<qstart, 0> -> q2 # here is another comment, comments will help the grader
<qstart, 1> -> q6 # understand your code
<q2, 0> -> q3
<q2, 1> -> q4
<q3, 0> -> q3
<q3, 1> -> q3
<q4, 0> -> q5
<q4, 1> -> q3
<q5, 0> -> q3
<q5, 1> -> q4
<q6, 0> -> q6
<q6, 1> -> q6
accept:
q4
q5
Problem 1

Give a NFA (or DFA) in the format described above for the following regular expression:

\[(00|11)^*10(11|00)^*10(0|1)^*\]

We provide you with a DFA/NFA syntax checker. Any syntax issues with this part of the assignment will result in zero credit.

![DFA/NFA Diagram]

Solution:

\(\text{sigma: } \#\text{Input Alphabet}\)

\(0\)

\(1\)

\(Q: \#\text{States}\)

\(q_{\text{start}}\)

\(q_1\)

\(q_2\)

\(q_3\)

\(q_4\)

\(q_5\)

\(q_6\)

\(q_7\)

\(\text{delta: } \#\text{Transitions}\)

\(<q_{\text{start}}, 0> \rightarrow q_1\)

\(<q_{\text{start}}, 1> \rightarrow q_2\)

\(<q_1, 0> \rightarrow q_{\text{start}}\)

\(<q_1, 1> \rightarrow q_7\)

\(<q_2, 0> \rightarrow q_3\)

\(<q_2, 1> \rightarrow q_1\)

\(<q_3, 0> \rightarrow q_4\)

\(<q_3, 1> \rightarrow q_5\)

\(<q_4, 0> \rightarrow q_3\)

\(<q_4, 1> \rightarrow q_7\)

\(<q_5, 0> \rightarrow q_6\)
<q5, 1> -> q3
<q6, 0> -> q6
<q6, 1> -> q6
<q7, 0> -> q7
<q7, 1> -> q7
accept:
qu6
Problem 2

For every string \( w = w_1w_2 \ldots w_n \), the string written in reverse, denoted \( w^r \), is the string \( w_nw_{n-1} \ldots w_1 \). For any language (set) \( L \), let \( L^R = \{ w^r \mid w \in L \} \). Show that if \( L \) has a DFA then \( L^R \) has a DFA.

Solution: Suppose \( L \) is recognizable by a DFA \( D \). Construct a non-deterministic finite automata \( N \) via the following procedure:

- If \( q \) is a start state in \( D \), then \( q \) is an accepting state in \( N \)
- If \( \langle q_i, s \rangle \to q_j \) is a transition in \( D \), then \( \langle q_j, s \rangle \to q_i \) is a transition in \( N \)
- If \( q \) is an accepting state in \( D \), then \( q \) is a start state in \( N \)
  (i.e. there is an extra state \( q^* \) in \( N \) with epsilon transitions to all accepting states in \( D \), and \( q^* \) is the start of \( N \))
- \( N \) has no other transitions, states, start states, or accepting states

We now show that \( L(N) = L^R \) by proving that, for any \( w, w \in L^R \iff w \in L(N) \).

By construction:

\[ w \in L^R \iff w^r \in L \]

\[ \iff w^r \text{ is accepted by } D \]

\[ \iff \text{there is some sequence of states } S = q_1, q_2, q_3, \ldots, q_k \text{ such that} \]

\[ - q_1 \text{ is the start state of } D \]

\[ - q_k \text{ is an accepting state of } D \]

\[ - \text{for every adjacent pair } (q_i, q_j) \text{ in } S \text{ there is a transition } \langle q_i, w_i^r \rangle \to q_j \text{ in } D \]

\[ \iff \text{there is some sequence of states } S' = q_k, q_{k-1}, q_{k-2}, \ldots, q_1 \text{ such that} \]

\[ - q_k \text{ is a start state of } N \]

\[ - q_1 \text{ is the accepting state of } N \]

\[ - \text{for every adjacent pair } (q_i, q_j) \text{ in } S' \text{ there is a transition } \langle q_i, w_i \rangle \to q_j \text{ in } N \]

\[ \iff w \text{ is accepted by } N \]

\[ \iff w \in L(N) \]

Since every non-deterministic finite automata is equivalent to a deterministic one, there is some DFA that recognizes \( L^R \).
Problem 3

Prove that the given language, L, defined below, is regular by defining a DFA in the format described above.

Let \( \Sigma = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \)

When given a string \( w \in \Sigma^* \) consider the top row as a number written in binary and consider the bottom row a second number written in binary.

So for example the string \( w = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) represents the number 5 on top and 2 on the bottom.

\( L = \{ w \mid w \in \Sigma^* \text{ and the top binary number in } w \text{ is greater than the bottom number in } w \} \)

To represent the tile symbols from \( \Sigma \) in your text file, let \( \begin{bmatrix} 0 \\ 0 \end{bmatrix} \) be 00, \( \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) be 01, \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) be 10, \( \begin{bmatrix} 1 \\ 1 \end{bmatrix} \) be 11.

We provide you with a DFA/NFA syntax checker. Any syntax issues with this part of the assignment will result in zero credit

![DFA Diagram]

Solution:

**sigma:** #Input Alphabet

00
01
10
11

**Q:** #States

qstart
q1
q2

**delta:** #Transitions

<qstart, 00> -> qstart
<qstart, 01> -> q2
<qstart, 10> -> q1
<qstart, 11> -> q1

11, 00

01

11, 10, 01, 00

2

11, 10, 01, 00

10

1

11, 10, 01, 00

start
<qstart, 11> → qstart
<q1, 00> → q1
<q1, 01> → q1
<q1, 10> → q1
<q1, 11> → q1
<q2, 00> → q2
<q2, 01> → q2
<q2, 10> → q2
<q2, 11> → q2
accept:
q1
Problem 4

For languages $A$ and $B$ let the *shuffle* of $A$ and $B$ be the language $L$,

$$L = \{ w \mid w = a_1b_1a_2b_2\ldots a_kb_k, \text{ where } a_1, a_2, \ldots, a_k \in A, \text{ and } b_1, b_2, \ldots, b_k \in B, \text{ and each } a_i, b_j \in \Sigma^* \}$$

Prove that if $A$ and $B$ are regular then $L$ is regular.

Solution: Suppose $A$ and $B$ are regular. Then $A$ and $B$ can be represented as regular expressions $R_A$ and $R_B$. We define $R_L = (R_AR_B)^*$ and show that $R_L = L$.

$$w \in L \iff w = a_1b_1a_2b_2\ldots a_kb_k, \text{ where } a_1, a_2, \ldots, a_k \in A \text{ and } b_1, b_2, \ldots, b_k \in B, \text{ for some } k$$

$$
\iff w = a_1b_1a_2b_2\ldots a_kb_k, \text{ where } a_1, a_2, \ldots, a_k \in R_A \text{ and } b_1, b_2, \ldots, b_k \in R_B
$$

$$
\iff w \in (R_AR_B)(R_AR_B)\ldots(R_AR_B)_k
$$

$$\iff w \in (R_AR_B)^*
$$

Since $R_L$ is a regular expression, and $R_L = L$, $L$ is regular.
Problem 4’

For languages $A$ and $B$ let the \textit{shuffle} of $A$ and $B$ be the language $L'$,

$$L' = \{ w \mid w = a_1 b_1 a_2 b_2 \ldots a_k b_k, \text{ where } 'a_1 a_2 \ldots a_k' \in A, \text{ and } 'b_1 b_2 \ldots b_k' \in B \}$$

Prove that if $A$ and $B$ are regular then $L'$ is regular.

Solution: We first use dfa’s for $A$ and $B$ to construct a dfa $D_S$ to recognize the language

$$S_L = \{ w \mid w = (a_1, b_1)(a_2, b_2)\ldots (a_k, b_k) \text{ s.t. } 'a_1 a_2 \ldots a_k' \in A \text{ and } 'b_1 b_2 \ldots b_k' \in B \}$$

where $(a_j, b_k)$ represents a single character in the alphabet. We then use $D_S$ to construct an nfa $N_L$ to recognize $L'$.

Suppose $A$ and $B$ are regular. Then there are dfa’s

$$D_A = (\Sigma_A, Q_A, q_0A, \delta_A, F_A) \quad \text{and} \quad D_B = (\Sigma_B, Q_B, q_0B, \delta_B, F_B)$$

which recognize $A$ and $B$ respectively. Define

$$D_S = (\Sigma_S, Q_S, q_0S, \delta_S, F_S) \quad \text{where}$$

$$\Sigma_S = \Sigma_A \times \Sigma_B$$

$$Q_S = Q_A \times Q_B$$

$$q_0S = (q_0A, q_0B)$$

$$F_S = F_A \times F_B \quad \text{and} \quad \delta_S \text{ is defined as follows:}$$

$$\langle (q_i, q_s), (j, t) \rangle \rightarrow (q_k, q_u) \in \delta_S \iff \langle q_i, j \rangle \rightarrow q_k \in \delta_A \text{ and } \langle q_s, t \rangle \rightarrow q_u \in \delta_B$$

So, by construction, for arbitrary $k$

$$w \in L' \iff w = a_1 b_1 a_2 b_2 \ldots a_k b_k \text{ where } 'a_1 a_2 \ldots a_k' \in A \text{ and } 'b_1 b_2 \ldots b_k' \in B$$

$$\iff w' \in S_L \text{ where } w' = (a_1, b_1)(a_2, b_2)\ldots (a_k, b_k)$$

$$\iff \text{ in both } D_A \text{ and } D_B, \text{ there are accepting sequences of states which consume } a \text{ and } b \text{ respectively}$$

$$\iff \text{ there is an accepting sequence of states in } D_S \text{ which consumes } w'$$

$$\iff w' \in L(D_S)$$

where this last step holds (in both directions) because $a$ and $b$ are required to be the same length, and because $D_A$ and $D_B$ are dfa’s, so their accepting sequences have to line up for any string in the language.

We now construct an nfa $N_L = (\Sigma_L, Q_L, q_0S, \delta_L, F_S)$ to recognize $L'$. $\Sigma_L = \Sigma_A \cup \Sigma_B$. $Q_L$ is identical to $Q_S$, except that $Q_L$ contains one additional state for each transition in $\delta_S$. $\delta_L$ is defined as follows: for each transition in $\delta_S$ of the form

\[
\begin{array}{c}
1 \quad \text{ab} \quad 2
\end{array}
\]

there are transitions in $\delta_L$ of the form:

\[
\begin{array}{c}
1 \quad a \quad * \quad b \quad 2
\end{array}
\]
where (1) and (2) are any (possibly identical) states, and (\(\ast\)) is the new state for that particular \(\delta_S\) transition. There are no other transitions in \(\delta_L\).

By construction,

\[
\begin{align*}
w' & \in L(D_S) \iff w' = (a_1, b_1)(a_2, b_2)\ldots(a_k, b_k) \quad \text{and there is some accepting sequence of } D_S \text{ that consumes } w \\
& \iff \text{there is some accepting sequence of } N_L \text{ that consumes } w, \text{ where} \\
& w = a_1b_1a_2b_2\ldots a_kb_k \\
& \iff w \in L(N_L)
\end{align*}
\]

Since

\[
w \in L' \iff w' \in L(D_S) \iff w \in L(N_L)
\]

\(L'\) is recognized by an nfa. Therefore, \(L'\) is regular.
*Note that, in the following, we use \( a \circ b \) to denote concatenation. For example, if \( a = '000' \) and \( b = '11' \), then \( a \circ b = '0011' \).

**Problem 5**

Prove that for every \( k > 1 \), a language \( A_k \subseteq \{0, 1\}^* \) exists that can be recognized by a DFA with \( k \) states, but not by one with \( (k - 1) \) states.

**Solution:** Let \( s^{(x)} \) represent a string of \( x \) ’s, or \( \underbrace{s \ldots s}_{x} \). Define \( A_k = \{0^{(k-1)}\} \). We construct a DFA with \( k \) states that recognizes \( A_k \), then show that no DFA with \( k - 1 \) states recognizes \( A_k \).

Let \( D_k \) be the following DFA:

```
1 ----> 0 ----> 2 ----> 0 ----> 3 ----> 0 ----> \ldots ----> 0 ----> k-1 ----> 0 ----> k
```

We can see that \( w \) is accepted by \( D_k \) iff \( w \) is a sequence of \( k - 1 \) ‘0’s iff \( w \in A_k \). Therefore, every \( A_k \) is recognizable by some DFA with \( k \) states.

We show by induction on \( k \) that \( A_k \) is recognized by no DFA with \( k - 1 \) states.

**Base case:** \( k = 2 \). \( A_2 = \{1\} \). So, a DFA that recognizes \( A_2 \) must accept the string ‘1’ and reject everything else. Let \( D \) be a DFA with 1 state \( q \). If \( q \) is an accept state, then \( D \) accepts on the empty string, and so does not recognize \( A_2 \). If \( q \) is a reject state, then \( D \) rejects on all strings, and so does not recognize \( A_2 \). Therefore, \( A_2 \) is recognized by no single state DFA.

**Inductive step:** suppose that, for some \( n \geq 2 \), \( A_n \) is recognized by no DFA with \( n - 1 \) states. Suppose, for the sake of contradiction, that \( A_{n+1} = \{0^{(n)}\} \) is recognized by some DFA \( D \) with \( n \) states. Then there is some accepting sequence of states \( S = q_1, q_2, \ldots, q_{n-1}, q_n \) that represents the computation of \( 0^{(n)} \) in \( D \). In fact, since \( D \) is a DFA that recognizes \( A_{n+1} \), \( S \) is the only accepting sequence of \( D \).

We can see that \( S \) can contain no repeats (if it did, it would contain a loop, in which case we could construct a string \( w' \notin A_{n+1} \) that is accepted by \( D \)). So, each \( q_i \in S \) is distinct. We construct a machine \( D' \) as follows: \( D' \) is identical with \( D \) except that \( q_n \) has been removed and \( q_{n-1} \) is the new accept state. We claim that \( D' \) is a DFA with \( n - 1 \) states that recognizes \( A_n \).

Since \( D \) contains no loops, the sequence \( S' = q_1, q_2, \ldots, q_{n-1} \) is an accepting computation for \( 0^{(n-1)} \) in \( D' \). Suppose \( D' \) accepts some other string \( z \neq 0^{(n-1)} \). Then, by construction of \( D' \), \( D \) accepts some string \( z \circ '0' \neq 0^{(n)} \), and so does not recognize \( A_{n+1} \). This contradicts our supposition, so \( D' \) recognizes \( A_n \). However, by the inductive hypothesis, there is no \( n - 1 \) state DFA that recognizes \( A_n \), so there is no \( n \) state DFA that recognizes \( A_{n+1} \).

Therefore, the result is proved.